Comparing YOLOv5, YOLOv7 and YOLOv8 for Autonomous License Plate Recognition System

Karen Sunaryo[1], Gilbert Lorentz[2], Jessica Darmawan[3]

[123]*Data Science and Analytics, Petra Christian University, Surabaya, Jawa Timur 60234, Indonesia*

[1]c14210248@john.petra.ac.id

[3]c14210192@john.petra.ac.id

[3]c14210045@john.petra.ac.id

Abstract—**Addressing the challenges of computational complexity and inflexibility in traditional License Plate Recognition (LPR) methods, we propose an end-to-end deep learning model using the YOLO (You Only Look Once) framework for enhanced license plate recognition in natural scenarios. This project focuses on developing a robust LPR system by experimenting with various YOLO versions, including YOLOv8, YOLOv7, and YOLOv5, to identify the most accurate model for recognizing Indonesian license plates. First, we trained the model to accurately detect and crop license plates from images. Next, we implemented preprocessing steps to enhance image quality, thereby improving the accuracy and efficiency of character recognition within the license plates. The goal of the evaluation of various YOLO versions was to identify the best model for the LPR system that was especially made for Indonesian license plates, which come in a variety of colors, including red, white, and black. The licence plate recognition model achieved 96.85% accuracy while licence plate character recognition models that were compared by quantitive measures in detecting and recognizing character show that xyz.**

Keywords— **YOLO, License Plate Recognition, Deep Learning, Indonesian License Plate, Object Recognition, Characters Recognition, Autonomous**

I. Introduction

License plates play a crucial role in the identification and tracking of vehicles, serving as unique identifiers that link vehicles to their owners. This is essential for various applications, including law enforcement, toll collection, parking management, and traffic monitoring. In recent years, there has been a surge in the adoption of License Plate Recognition (LPR) technology, which allows for quick and automated identification of vehicle license plates. LPR systems have become indispensable tools for enhancing security, improving traffic flow, and streamlining operations in busy areas such as toll booths and parking facilities. The widespread use of LPR technology underscores its importance in modern transportation and urban management, providing a reliable and efficient means of vehicle identification.

The technology of License Plate Recognition (LPR) currently faces significant challenges due to the complexity of its processes. Traditional LPR methods require extensive computational time and involve complex steps of segmentation, character detection, and object recognition of license plates [1]. This complexity leads to delays in license plate recognition, hindering system performance, especially in situations where rapid recognition is crucial, such as at toll gates or busy public parking areas. Additionally, LPR technology still exhibits limitations in terms of flexibility. Although LPR systems are increasingly used, they remain inflexible, requiring vehicles to be positioned in specific ways under certain conditions for accurate detection. This inflexibility poses difficulties, particularly for complex or real-time scenarios. Beyond complexity and flexibility issues, resistance to noise is another critical focus. A reliable LPR system must effectively handle noise in vehicle images or videos caused by external factors such as weather and lighting conditions.

The rapid advancement of deep learning theory and technology in recent years has led to tremendous progress in feature extraction, image representation, classification, and recognition for the object detection technique based on deep understanding [2]. In addition to enhancements in data processing, network architecture, loss function, and other areas, researchers have put forth a number of methods for the object detection algorithm. Among these, YOLO (You Only Look Once), introduced by Redmon et al. in 2015, stands out for its real-time processing capabilities and high accuracy. YOLO has been continuously refined, with versions like YOLOv5, YOLOv7, and YOLOv8 pushing the boundaries of speed and accuracy, making it a leading choice in various applications including autonomous driving and surveillance [3].

The core of the YOLO target detection algorithm lies in the model's small size and fast computation speed. The YOLO structure is straightforward. Using a neural network, it can output the bounding box's category and position directly. YOLO's speed is fast because all it takes to obtain the final detection result is to transfer the picture to the network [4]. The efficiency of the YOLO algorithm has been well-documented in various studies. For example, in a study by Redmon et al., the authors compared the performance of YOLO with other object detection algorithms and found that YOLO achieved faster processing times while maintaining competitive accuracy levels, underscoring the advantage of its direct output approach and streamlined neural network architecture [5]. This further supports the notion that YOLO's speed is a result of its direct output approach and streamlined neural network architecture. Additionally, research by Wang et al. [6] delves into the advancements made in YOLO's model architecture, particularly in terms of improving its accuracy without sacrificing speed. By incorporating techniques such as feature pyramid networks and attention mechanisms, YOLO has been able to enhance its detection capabilities while remaining computationally efficient.

Numerous versions of YOLO have been created over time, including YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7 and YOLOv8. The possibilities for real-time object detection are still being expanded by these developments. Better handling of small objects, improved training procedures, and the addition of new activation functions like ELU (Exponential Linear Units), which boost model performance, are some of the improvements [3, 7]. Among the various iterations, YOLOv5, YOLOv7, and YOLOv8 are distinguished by their outstanding performance and advancements. YOLOv5 marks a significant advancement where it offers a lighter model weight compared to YOLOv4, leading to increased accuracy, reduced computational load, and enhanced detection speed [8]. Additionally, YOLOv5 is adept at detecting smaller objects through augmented backbone layers and larger input sizes, resulting in improved capability to detect objects of varying scales. YOLOv7 presents several notable advantages and superiorities over previous versions and other object detection models. Firstly, it achieves exceptional accuracy and speed performance, outperforming established counterparts like YOLOv4, YOLOX, YOLOR, Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, and ViT-Adapter-B [9]. YOLOv7 also implements a comprehensive scaling strategy, augmenting model depth, breadth, and resolution to cater to diverse application needs. Its re-parameterization strategy and RepConvN integration further enhance its robustness and efficiency, contributing to its impressive real-time object detection capabilities, faster convergence. YOLOv7 stands out as a top-performing object detection model, excelling in accuracy, speed, and versatility [10]. While YOLOv8 distinguishes itself with higher precision, recall, and mean average precision, demonstrating superior performance in detecting pulmonary carcinoma compared to YOLOv7. The model's improved accuracy, particularly in identifying small objects, is a significant advantage for early disease detection in medical imaging. This enhanced capability is primarily attributed to YOLOv8's anchor-free design, which optimizes the detection process and enhances reliability [11]. Therefore, the main contributions of this paper are as follows:

- We present a comprehensive analysis of various versions of the YOLO (You Only Look Once) object detection algorithm to determine which version performs best in recognizing Indonesia's vehicle license plates.
- We explore the performance of License Plate Recognition (LPR) systems under special circumstances, including scenarios where license plates are obstructed, too small, or have unique characteristics.

To achieve these contributions, we will utilize YOLOv5, YOLOv7, and YOLOv8 to analyze three variations of Indonesia's vehicle license plates, distinguished by their colors: red, white, and black. The models will be tested on images exhibiting special characteristics, such as plates that are too small, tilted, or captured with flash, to thoroughly evaluate their performance under challenging conditions. This approach will allow us to identify the strengths and weaknesses of each YOLO version in diverse real-world scenarios.

II. RELATED WORKS

Several studies have explored the use of YOLO (You Only Look Once) for LPR. Papers like [1,3] demonstrate the effectiveness of YOLO in real-time license plate detection. [3] specifically focuses on improving YOLOv5 for LPR by incorporating a Gated Recurrent Unit (GRU) for better character recognition. Furthermore, next research [8] explores improvements on Faster R-CNN, another popular object detection model, for small object detection like license plates. This highlights the ongoing effort to optimize detection for specific tasks. Besides that, previous work [6,9] showcase the continuous development of YOLO. They present newer versions (YOLOv4, YOLOv7) that achieve better accuracy and speed compared to previous iterations.

While YOLO has shown promise in License Plate Recognition (LPR) tasks, there are areas where existing research can be improved. One key weakness lies in the issue of dataset specificity. YOLO's performance can be significantly impacted by variations in factors like license plate formats, lighting conditions, and overall image quality. Unfortunately, not all studies address this by training their models on diverse datasets. This lack of data variety can limit the robustness of the LPR system in real-world scenarios where these variations are commonplace.

Another challenge facing LPR systems is the inherent trade-off between real-time performance and accuracy. Achieving high accuracy often requires processing power that may not be feasible in resource-constrained environments. Optimizing YOLO for faster processing, particularly for situations with limited computational resources, remains an ongoing area of research. Addressing this trade-off is crucial for expanding the practical applications of LPR systems.

Finally, some existing research focuses on improving specific versions of YOLO, such as study [3] which explores enhancements for YOLOv5. While valuable, there's a potential benefit in exploring the effectiveness of newer iterations like YOLOv7 on LPR tasks. Investigating these advancements could lead to improved performance and address some of the limitations of earlier YOLO versions in the context of LPR. By tackling these weaknesses, future research can contribute to the development of more robust, efficient, and adaptable LPR systems using YOLO. Therefore, this paper proposes an in-depth study that not only benchmarks the performance of different YOLO versions but also provides insights into their strengths and weaknesses in the context of LPR. By focusing on real-world scenarios and challenging conditions, our research aims to identify the most suitable YOLO model for effectively and accurately recognizing Indonesia's vehicle license plates.

III. METHODOLOGY

A. Model Framework

The proposed license plate recognition system comprises two key components: license plate detection and license plate character recognition, which together form the network model framework illustrated in Figure 1. License Plate Detection model will be trained using YOLOv8 framework while License Plate Character Recognition model will be trained with YOLOv5, YOLOv7 and YOLOv8. Following each detection step, Non-Maximum Suppression (NMS) will be applied to remove duplicate or overlapping bounding boxes. Each model will undergo training on a specific dataset tailored to its functionality and application context. This approach ensures robust performance and accuracy in identifying and interpreting license plates within diverse real-world scenarios.

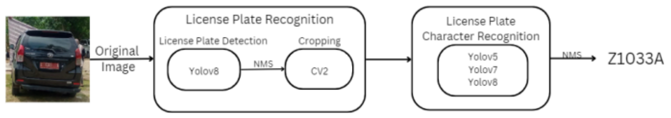


Fig 1. License Plate Recognition Model Framework

B. *YOLOv5*

YOLOv5, developed by Ultralytics, represents a significant advancement in the YOLO series, particularly in real-time object detection. Although YOLOv5 has not been published in peer-reviewed literature, it has gained widespread usage due to its effectiveness and ease of deployment [14, 15]. Implemented using PyTorch, YOLOv5 operates at an inference speed of 140 frames per second, making it suitable for real-time applications. The model comes in five variants, nano, small, medium, large, and extra-large [13, 18], for each tailored to specific dataset requirements and application contexts.

The architecture of YOLOv5 contains a backbone network for feature extraction, which uses CSPNet [12, 16] with Darknet. The neck component comprises multi-scale detection algorithms such as Spatial Pyramid Pooling (SPP) and Path Aggregation Network (PANet) [12, 17]. PANet improves the propagation of low-level features and uses adaptive feature pooling to link feature grids with all feature levels, ensuring useful information at each feature level is directly propagated to subsequent subnetworks. The model's brain makes final predictions, such as bounding boxes, objectness scores, and class probabilities over many scales, in order to properly handle different size objects. CSPNet tackles the issue of recurring gradient information in large-scale backbones by incorporating gradient changes into the feature map, which reduces parameters and FLOPS. The head of YOLOv5, known as the YOLO layer, generates feature maps in three different sizes: 18 × 18, 36 × 36, and 72 × 72. This multi-scale prediction capability enables the model to effectively detect small, medium, and large objects. This approach, which originates from YOLOv3, enhances YOLOv5's ability to handle objects of varying sizes by leveraging the benefits of multi-scale prediction [12]. This enables faster inference, more accuracy, and a smaller model size.

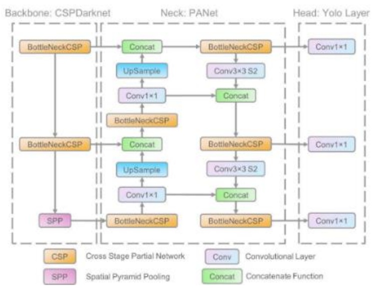


Fig 2. YOLOv5 Architecture [12]

YOLOv5 ideal for applications requiring immediate feedback, such as surveillance and autonomous driving. However, while one-stage detectors are faster, two-stage detectors like Faster R-CNN often achieve higher accuracy [19], making them preferable for tasks where precision is crucial. YOLOv5 v7.0 [18] introduced state-of-the-art instance segmentation models, trained on the COCO dataset, demonstrating superior accuracy and speed, surpassing current benchmarks. These models are easy to train, validate, and deploy, with features like Paddle Paddle export, auto-caching, and Comet logging for interactive visualization and debugging.

C. *YOLOv7*

One of the key innovations in YOLOv7 is the introduction of Extended Efficient Layer Aggregation Networks (E-ELAN). This architecture focuses on improving the learning capabilities of the network without altering the original gradient path. E-ELAN utilizes group convolution to increase the cardinality of features, and combines these features in a shuffle and merge manner, which enhances the diversity and utility of the learned features. This approach allows the network to handle more complex and diverse datasets while maintaining computational efficiency [21, 22].

Additionally, YOLOv7 employs a compound scaling method tailored for concatenation-based models. Traditional model scaling methods adjust parameters such as resolution, depth, and width independently. However, for concatenation-based architectures like YOLOv7, scaling depth alone can alter the input and output channel ratios, potentially degrading model performance. The compound scaling method in YOLOv7 scales both depth and width in a coordinated manner, ensuring the model retains its optimal structure and efficiency across various scales [22].

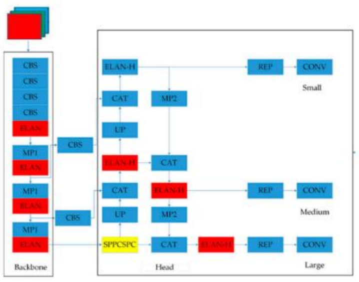


Fig 3. YOLOv7 Architecture [20]

Another significant feature of YOLOv7 is its Trainable bag-of-freebies approach, which includes techniques like planned re-parameterized convolution. This technique adapts convolution layers to different network architectures by analyzing gradient flow paths, thereby optimizing the placement of re-parameterized convolutions. This results in better integration with residual and concatenation connections, ultimately enhancing the model's performance without adding inference complexity [22]. YOLOv7 also introduces a novel label assignment strategy called Coarse for auxiliary and fine for lead head [22]. This method uses an auxiliary head in conjunction with the primary head to improve label assignment during training. The auxiliary head refines coarse labels, which are then fine-tuned by the lead head, leading to more accurate object detection across various scales and object types.

D. *YOLOv8*

YOLOv8 is a leading-edge and state-of-the-art model that builds upon the improvements and innovations of the previous YOLOv5 [25]. YOLOv8 incorporates innovations such as Extended Efficient Layer Aggregation Networks (E-ELAN), which improve learning capabilities without altering gradient paths. This enhancement helps the model handle more complex datasets while maintaining efficiency [24]. YOLOv8 is not just limited to object detection; it also excels in tasks like instance segmentation, image classification, and even human pose estimation, making it a versatile tool for various computer vision applications [24, 25]. YOLOv8 have high accuracy and speed in detecting objects in varied and challenging environments [26].

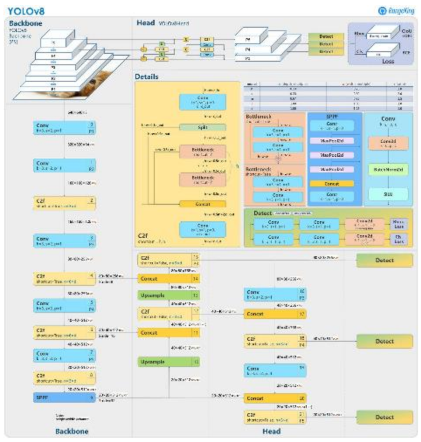
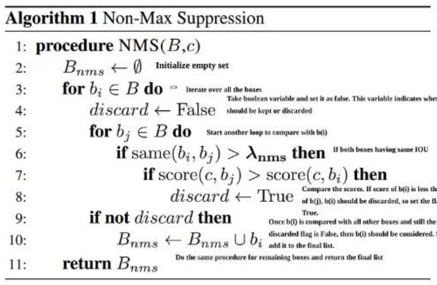


Fig 4. YOLOv8 Architecture [23]

YOLOv8 also uses a new method for label assignment during training called Coarse for auxiliary and fine for lead head, improving the accuracy of detecting objects across various scales and types [25]. The YOLOv8 documentation is comprehensive and well-organized, covering everything from installation and configuration to advanced usage scenarios. This helps both novice and experienced users maximize the potential of YOLOv8 [26]. YOLOv8 is ideal for a wide range of applications, including surveillance and security for real-time object detection, enhancing autonomous driving with the detection and classification of objects in the vehicle's environment, retail analytics for inventory management, and healthcare for medical image analysis [27].

E. *Non-Maximum Suppression (NMS)*

For nearly 50 years, NMS has been a key component of many detection methods in computer vision. It was first applied to edge detection algorithms [29]. Subsequently, it has been applied to multiple tasks like feature point detection [30], face detection [31] and object detection [32]. In edge detection, NMS performs edge thinning to remove spurious responses [29]. NMS is effective in performing local thresholding to obtain unique feature point detections, it is performed by partitioning bounding boxes into disjoint subsets using an overlap criterion.



Algorithm 1 Non-Max Suppression

```
1:  procedure NMS($B$,$c$)
2:      $B_{nms} \leftarrow \emptyset$   Initialize empty set
3:      for $b_i \in B$ do   => Iterate over all the boxes
4:          $discard \leftarrow$ False   Take boolean variable and set it as false. This variable indicates whether b(i) should be kept or discarded
5:          for $b_j \in B$ do   Start another loop to compare with b(i)
6:              if same$(b_i, b_j) > \lambda_{nms}$ then   If both boxes having same IOU
7:                  if score$(c, b_j) >$ score$(c, b_i)$ then
8:                      $discard \leftarrow$ True   Compare the scores. If score of b(i) is less than that of b(j), b(i) should be discarded, so set the flag to True.
9:          if not $discard$ then   Once b(i) is compared with all other boxes and still the discarded flag is False, then b(i) should be considered. So add it to the final list.
10:             $B_{nms} \leftarrow B_{nms} \cup b_i$
11:     return $B_{nms}$   Do the same procedure for remaining boxes and return the final list
```

Fig 5. Non-Max Suppression Algorithm [28]

Non-Maximum Suppression (NMS) is an essential post-processing technique widely used in object detection algorithms, including the YOLO (You Only Look Once) series. The primary purpose of NMS is to eliminate redundant bounding boxes that may have been predicted for the same object, ensuring that only the most accurate and relevant bounding box is retained for each detected object.

This redundancy occurs because the model tries to account for various object sizes and aspect ratios, which leads to multiple overlapping predictions. NMS addresses this issue by sorting these bounding boxes based on their confidence scores in descending order. The algorithm then selects the bounding box with the highest confidence score and suppresses all other bounding boxes that overlap significantly with the selected box, usually measured by the Intersection over Union (IoU) metric. This process is repeated iteratively until all boxes have been either selected or discarded based on the IoU threshold, typically set at 0.5 [28].

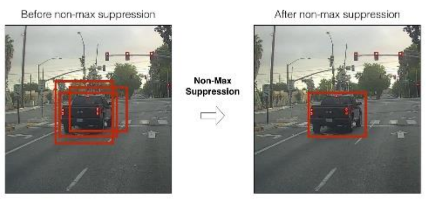


Fig 6. Detection using Non-Max Suppression [28]

IV. EXPERIMENTS AND RESULT ANALYSIS

A. *Dataset*

The license plate dataset used for the training and experiments in this paper is the open source Roboflow Dataset of Indonesia License Plate and Vehicle images. For license plate recognition stage, we used plat-nomor-indo and ocrv2 dataset with totalling 4823 images. The images in plat-nomor-indo are 4000x3000 in size while images in ocrv2 are variative in size. Both datasets contain license plate and vehicle image in many complex environments and positions. Some images are shot from above, resulting in varied tilts. Others are taken on the road, making them blurry, too bright, or other challenging conditions. Part of the sample data is shown in the Fig.7.

For character recognition stage, we used kombinasi v1 and carplatedetection-ocr dataset with totalling 7546 images. The images in kombinasi v1 are 640x640 in size while images in carplatedetection-ocr are variative in size. Both datasets contain license plate and vehicle image in many complex environments and positions. Some images are shot from above, resulting in varied tilts. Others are blurry, too bright, or other challenging conditions. Part of the sample data is shown in the Fig.8.

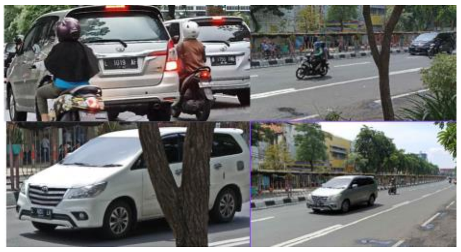


Figure 7. License Plate Recognition sample data

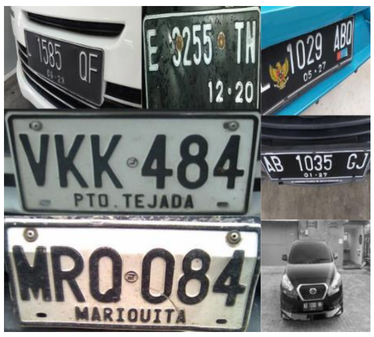


Figure 8. License Plate Character Recognition sample data

B. *Evaluation Indicators*

License Plate Recognition model will be tested on the images shown in Fig 9. For each correct recognisation the model will earn 0.44 points. The OCR model will be tested on the images shown in Fig 10. Each image will serve as a basis for evaluating the model's performance through two main criteria. Firstly, the model will be assessed on its ability to detect characters accurately, with each correct detection earning 8.125 points per image. Secondly, the model's capability to classify the detected characters correctly will be evaluated, with each correct classification earning 4.375 points per image. These indicators are designed to comprehensively measure the model's effectiveness in handling license plate images that present various challenges, such as varied angles and lighting conditions.

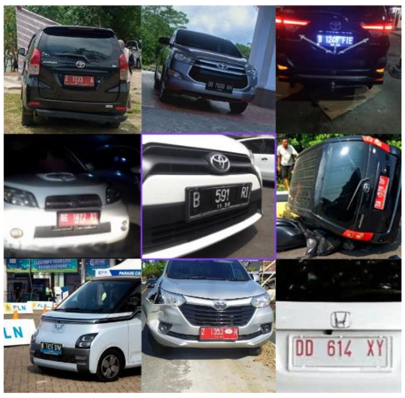


Figure 9. License Plate Recognition sample testing data

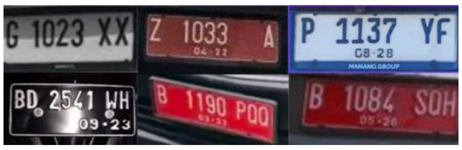


Figure 10. License Plate Character Recognition sample testing data

C. *Results Analysis*

1) License Plate Detection Model

In the training phase, we utilized pretrained weights for YOLOv8, as referenced in [33], and conducted additional training using a merged dataset comprising Plat-Nomor-Indo and OCRv2 datasets. The training process spanned 25 epochs to optimize model performance. The model architecture employed was YOLOv8n, a variant with 225 layers. We trained with a batch size of 8 using the AdamW optimizer with a learning rate of 0.002 and momentum of 0.9. Augmentations included blur, median blur, grayscale conversion, and contrast-limited adaptive histogram equalization (CLAHE) to enhance model robustness. The initial learning rate (lr0) was set to 0.01 but was automatically adjusted by the optimizer (optimizer=auto). Additionally, weight decay was applied at 0.0005 for certain parameters as part of the optimization strategy automatically determined by the optimizer configuration (optimizer=auto).

YOLOv8 model trained on the merged Plat-Nomor-Indo and OCRv2 dataset for 25 epochs achieved significant performance metrics, demonstrating high precision and recall rates for license plate detection. Specifically, as seen in Fig 11, the model achieved a precision of 96.66%, recall of 92.96%, and a mean Average Precision (mAP) of 97.49% at IoU (Intersection over Union) threshold 0.5 (mAP@50). The broader range mAP (mAP@50-95) was computed as 71.75%, indicating consistent performance across varying IoU thresholds. The results are summarized in the following metrics: precision(B) = 96.66%, recall(B) = 92.96%, mAP50(B) = 97.49%, and mAP50-95(B) = 71.75%. These metrics underscore the effectiveness of the training strategy and the robustness of the model in accurately detecting license plates.

Furthermore, the inference speed was observed to be efficient, with preprocessing taking approximately 1.06 seconds per image and the inference and postprocessing combined taking around 4.38 and 4.83 seconds per image, respectively.

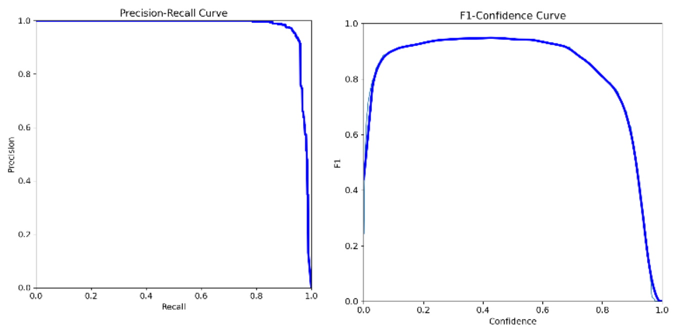


Figure 11. License Plate Detection F1 and PR curve

Further testing of the model included evaluating its performance on a diverse set of car images under various conditions, totaling 44 test cases. To increase the accuracy of the model, we applied Non-Maximum Suppression (NMS), which effectively removes duplicate bounding boxes and enhances detection precision. As shown in the comparison in Figure 12, the application of NMS resulted in a noticeable improvement, boosting the model's accuracy to 96.85%.

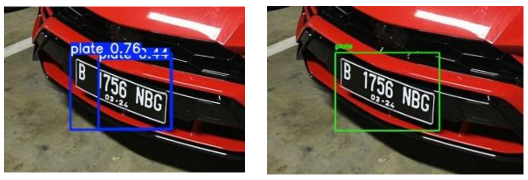


Figure 12. Yolov8 result before and after the usage of NMS

2) License Plate Character Recognition Model

a) YOLOv5

The models architecture, comprising 214 layers, facilitated comprehensive feature extraction and detection capabilities. With a batch size of 16, training efficiency was balanced with computational resources. The optimizer used was SGD, configured with a learning rate (lr) of 0.01, weight decay of 0.0005, and default momentum parameters, ensuring effective gradient descent during training. AutoAnchor functionality was crucial, dynamically adjusting anchor boxes with 3.48 anchors per target, thereby enhancing detection accuracy and achieving a high Best Possible Recall (BPR). We have created 6 models using kombinasi v1, carplatedetection-ocr and combinations of both. Table 1 shows the comparison of the models in detecting and identifying character in license plate.

TABLE I
YOLOv5 MODEL COMPARISON

Model	**Accuracy**
yolov5 1ds	53.75%
yolov5_merge_6	58.125%
yolov5_car25	62.5%
Yolov5_merge_6+7	71.25%
Yolov5_merge_6+7+7	83.125%
Yolov5_merge_6+7+7+7	58.75%

The YOLOv5_merge_6+7+7 model demonstrated the highest accuracy, achieving 83.125%. To further enhance this performance, we repeat the training for the final 7 epochs using specialized datasets. One dataset comprised black and white images, intended to improve the model's ability to detect and recognize features in monochromatic environments. The other dataset was resized to 640x640 dimensions, optimizing the model for consistent input size and potentially improving detection accuracy. After developing the model, we applied Non-Maximum Suppression (NMS) to the three models and tested them on vehicles with varied conditions and angles. The results of these tests are summarized in Table 2.

TABLE III
YOLOv5 ENHACED MODEL COMPARISON

Model	**Accuracy**	**Time**
YOLOv5_merge_6+7+7	27.66%	1.039 hours
YOLOv5_merge_6+7+7BW	31.91%	1.079 hours
YOLOv5_merge_6+7+7 resized	29.79%	0.945 hours

The F1-Confidence Curve and the Precision-Recall (PR) Curve, shown in Fig. 13, provide valuable insights into the performance of the YOLOv5 model. The F1-Confidence Curve shows that the F1 score peaks at 0.83 when the confidence threshold is around 0.395, indicating that this threshold offers the best balance between precision and recall for the model. The Precision-Recall curve further demonstrates the model's strong performance, achieving a mean Average Precision (mAP) of 0.864 at a threshold of 0.5. This high mAP score suggests that the model maintains a robust balance between precision and recall, effectively detecting objects across various thresholds. Additionally, the model is capable of identifying characters even when the license plates are tilted at a 45-degree angle and are blurry, as demonstrated in Fig. 14.

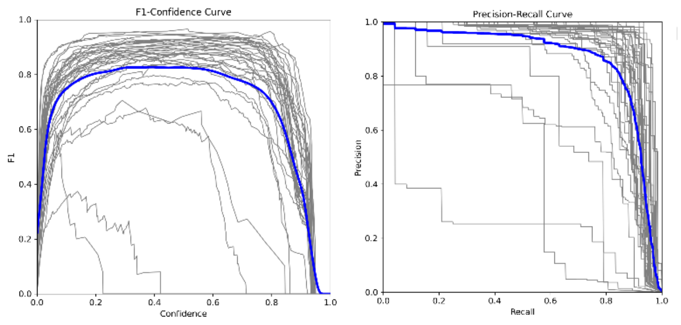


Figure 13. YOLOv5_merge_6+7+7BW F1 and PR curve

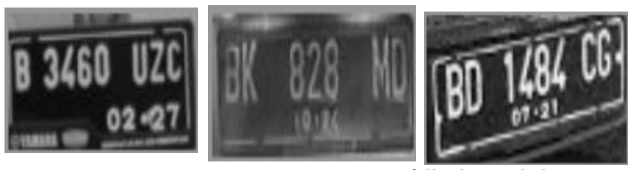


Figure 14. YOLOv5_merge_6+7+7BW successfully detected plate

b) YOLOv7

The training process of the models involved a batch size of 16 and spanned 6 or 8 epochs per training session, with the image size set to 640x640 pixels. Several hyperparameters were configured to fine-tune the training, including an initial learning rate of 0.01, momentum set to 0.937, and a weight decay of 0.0005. The training also incorporated warmup strategies to gradually introduce the model to the learning process, with 3 warmup epochs and specified warmup momentum and bias learning rates. Data augmentation techniques such as mosaic, mixup, and random flipping were employed to enhance the model's generalization capabilities. We have created 6 models using kombinasi v1, carplatedetection-ocr and combinations of both. Table 3 shows the comparison of the models in detecting and identifying character in license plate.

TABLE III
YOLOv7 MODEL COMPARISON

Model	**Accuracy**
Yolov7 1ds	58.125%
yolov7_car8	4.375%
yolov7_merge6	50.625%
yolov7_1ds_8+8	71.25%
yolov7_car8+8	16.875%
yolov7_1ds_8+8+8	55%

Among all YOLOv7 models, yolov7_1ds_8+8 has the best accuracy in identifying characters in the given license plate in various conditions and lightning. The PR and F1 curves, as shown in Fig.10, illustrate the performance of the YOLOv7_1ds_8+8 model. The precision-recall (PR) curve demonstrates the relationship between precision and recall, with the blue line representing the aggregated PR curve for all classes, achieving a mean Average Precision (mAP) of 0.751 at an IoU threshold of 0.5. This indicates that the model maintains a high level of precision and recall across all classes. The F1 score curve shows the F1 score as a function of the confidence threshold, with the peak F1 score of 0.73 occurring at a confidence threshold of 0.184. This peak suggests an optimal balance between precision and recall at this confidence level.

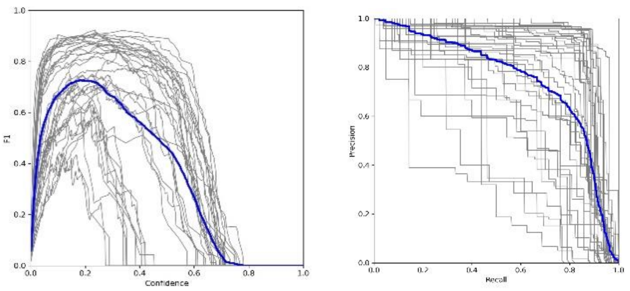


Figure 15. yolov7_1ds_8+8 F1 and PR curve

During testing on real vehicles under various angles and conditions, the model demonstrated poor performance with an accuracy of only 4.26%, significantly lower than the YOLOv5 model's 32% accuracy. This disparity suggests that YOLOv7's increased complexity may hinder its ability to accurately recognize characters on license plates, whereas the simpler architecture of YOLOv5 allows for better generalization and adaptation to real-world vehicle plate scenarios.

c) YOLOv8

The training environment for YOLOv8 emphasizes several key components to enhance model performance. The model architecture includes 225 layers with 3,017,868 parameters, focusing on efficient detection capabilities. Training leverages a variety of techniques such as Automatic Mixed Precision (AMP) for computational efficiency and dynamic augmentation strategies including blur, median blur, grayscale conversion, and contrast-limited adaptive histogram equalization (CLAHE) provided by Albumentations. The optimizer setup dynamically adjusts learning rates and momentums, optimizing for AdamW with a learning rate of 0.00025 and momentum of 0.9. Per training session will have 10-25 epochs. We have created 8 models using kombinasi v1, carplatedetection-ocr and combinations of both. Table 4 shows the comparison of the models in detecting and identifying character in license plate.

TABLE IV
YOLOv8 MODEL COMPARISON

Model	**Accuracy**
ocr_1ds	41.875%
10epoch_car	66.875%
1dr+25	46.875%
1ds+25+25	55%
1ds+25+1dsbyk10	66.875%
1ds+25+1dsbyk10+ds2	54.375%
merge10+10	58.125%
merge10+10+10	58.125%

Among all YOLOv8 models tested, the "1ds+25+1dsbyk10" and "10epoch_car" models achieved the highest overall accuracy at 66.875%. Specifically, for testing with vehicle pictures, the "10epoch_car" model outperformed all other models, achieving the best accuracy of 36.17%. This indicates that while both models excel in license plate only testing accuracy, the "10epoch_car" model is particularly effective in real-world vehicle detection scenarios. Notably, the model successfully predicted images with poor quality, including those that were blurred or angled 45 degrees clockwise or counterclockwise which can't be achieved by all the models which were shown in Fig. 16. The "10epoch_car" model performs better than other models trained with more datasets and epochs due to several key factors. By training for fewer epochs with a well-curated dataset, the "10epoch_car" model effectively avoids overfitting, maintaining a strong generalization capability. This approach ensures that the model learns robust features without becoming overly specialized to the training data, thereby enhancing its ability to perform well on unseen test data. Additionally, the model's simpler architecture and shorter training duration contribute to its efficiency in capturing essential features while minimizing noise and irrelevant patterns in the training data.

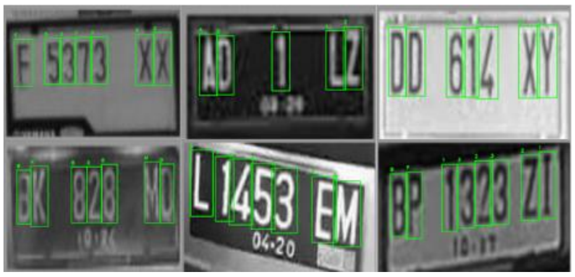


Figure 16. 10epoch_car successfully detected plate

V. Conclusions

In conclusion, the YOLOv8 model "10epoch_car" performs best in recognizing Indonesia’s vehicle license plates. The model demonstrates strong generalization capabilities, effectively handling images that are tilted, blurry, or captured with a flash. It maintains good performance even when license plates are crooked. However, the model has limitations with very small license plates, which become pixelated when zoomed in, or when plates are excessively tilted. To address these issues in future research, we plan to implement rotation correction for the images or randomly rotate a portion of the training dataset to improve the model’s ability to recognize tilted plates. Additionally, resizing and converting the training dataset to black and white could reduce model complexity and potentially enhance accuracy.

References

[1] P. Marzuki, A. R. Syafeeza, Y. C. Wong, N. A. Hamid, A. N. Alisa, and M. M. Ibrahim, "A design of license plate recognition system using convolutional neural network," Int. J. Electr. Comput. Eng. (IJECE), vol. 9, no. 3, pp. 2196-2204, 2019. [Online]. Available: https://doi.org/10.11591/ijece.v9i3.pp2196-2204

[2] S. A. Nawaz et al., "AI-based object detection latest trends in remote sensing, multimedia and agriculture applications," Frontiers in Plant Science, vol. 13, 2022. [Online]. Available: https://doi.org/10.3389/fpls.2022.1041514.

[3] H. Shi and D. Zhao, "License Plate Recognition System Based on Improved YOLOv5 and GRU," in IEEE Access, vol. 11, pp. 10429-10439, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3240439.

[4] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo Algorithm Developments," Procedia Computer Science, vol. 199, pp. 1066-1073, 2022. [Online]. Available: https://doi.org/10.1016/j.procs.2022.01.135.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788.

[6] Z. Wang, Y. Yuan, J. Hu, and H. Li, "YOLOv4: Optimal Speed and Accuracy of Object Detection," IEEE Access, vol. 9, pp. 29417-29429, 2021. https://doi.org/10.1109/ACCESS.2021.3052480.

[7] Z. Keita, "Yolo Object Detection explained: A beginner’s guide," DataCamp, 2022. Available at: https://www.datacamp.com/blog/yolo-object-detection-explained (Accessed: 08 June 2024).

[8] Cao, C., Wang, B., Zhang, W., Zeng, X., Yan, X., Feng, Z., Liu, Y., & Wu, Z. (2019). An Improved Faster R-CNN for Small Object Detection. IEEE Access, 7, 106838–106846. https://doi.org/10.1109/ACCESS.2019.2932731

[9] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.

[10] Kusuma, P. C., and B. Soewito, "Multi-object detection using YOLOV7 object detection algorithm on mobile device," Journal of Applied Engineering and Technological Science (JAETS), vol. 5, no. 1, pp. 305-320, 2023, doi: 10.37385/jaets.v5i1.3207.

[11] Elavarasu, M. and Govindaraju, K., "Unveiling the Advancements: YOLOv7 vs YOLOv8 in Pulmonary Carcinoma Detection," Journal of Robotics and Control (JRC), vol. 5, no. 2, pp. 459-470, 2024, doi: 10.18196/jrc.v5i2.20900

[12] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, “A forest fire detection system based on Ensemble Learning,” Forests, vol. 12, no. 2, p. 217, Feb. 2021. doi:10.3390/f12020217

[13] G. Jocher, A. Chaurasia, and A. Stoken, “Ultralytics/yolov5: Yolov5 in PyTorch > ONNX > CoreML > TFLite,” GitHub, https://github.com/ultralytics/yolov5 (accessed Jun. 29, 2024).

[14] S. S. Zaidi et al., “A survey of modern deep learning-based object detection models,” Digital Signal Processing, vol. 126, p. 103514, Jun. 2022. doi: 10.1016/j.dsp.2022.103514

[15] J. Solawetz, “Yolov5 v6.0 is here - new nano model at 1666 FPS,” Roboflow Blog, https://blog.roboflow.com/yolov5-v6-0-is-here/ (accessed Jun. 29, 2024).

[16] C.-Y. Wang et al., “CSPNet: A new backbone that can enhance learning capability of CNN,” 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Jun. 2020. doi:10.1109/cvprw50498.2020.00203

[17] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, “Panet: Few-shot image semantic segmentation with prototype alignment,” 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Oct. 2019. doi:10.1109/iccv.2019.00929

[18] “Ultralytics/yolov5: V7.0 - yolov5 sota realtime instance segmentation,” Zenodo, https://doi.org/10.5281/zenodo.7347926 (accessed Jun. 29, 2024).

[19] T. Mahendrakar et al., “Performance study of yolov5 and faster R-CNN for autonomous navigation around non-cooperative targets,” 2022 IEEE Aerospace Conference (AERO), Mar. 2022. doi:10.1109/aero53065.2022.9843537

[20] D. Yang et al., “Improved Yolov7 Network model for gangue selection robot for gangue and foreign matter detection in coal,” Sensors, vol. 23, no. 11, p. 5140, May 2023. doi:10.3390/s23115140

[21] C. Y. Wang, I. H. Yeh, and H. Y. M. Liao, “Wongkinyiu/Yolov7: Implementation of paper - yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” GitHub, https://github.com/WongKinYiu/yolov7 (accessed Jun. 29, 2024).

[22] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, “Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2023. doi:10.1109/cvpr52729.2023.00721

[23] L. Zhang, G. Ding, C. Li, and D. Li, “DCF-Yolov8: An improved algorithm for aggregating low-level features to detect agricultural pests and diseases,” Agronomy, vol. 13, no. 8, p. 2012, Jul. 2023. doi:10.3390/agronomy13082012

[24] S. Rath, “Yolov8 : Comprehensive guide to state of the art object detection,” LearnOpenCV, https://learnopencv.com/ultralytics-yolov8/ (accessed Jun. 30, 2024).

[25] “Yolov8 object detection model: What is, how to use,” Roboflow, https://roboflow.com/model/yolov8 (accessed Jun. 30, 2024).

[26] D. Reis, J. Kupec, J. Hong, and A. Daoudi, “Real-time flying object detection with yolov8,” arXiv.org, https://doi.org/10.48550/arXiv.2305.09972 (accessed Jun. 30, 2024).

[27] M. Sohan, T. Sai Ram, and Ch. V. Rami Reddy, “A review on Yolov8 and its advancements,” Data Intelligence and Cognitive Informatics, pp. 529–545, 2024. doi:10.1007/978-981-99-7962-2_39

[28] Sambasivarao. K, “non-maximum suppression (NMS),” Medium, https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c (accessed Jun. 30, 2024).

[29] A. Rosenfeld and M. Thurston, “Edge and curve detection for visual scene analysis,” IEEE Transactions on Computers, vol. C–20, no. 5, pp. 562–569, May 1971. doi:10.1109/t-c.1971.223290

[30] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, Nov. 2004. doi:10.1023/b:visi.0000029664.99615.94

[31] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of Simple features,” Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. doi:10.1109/cvpr.2001.990517

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” 2014 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2014. doi:10.1109/cvpr.2014.81

[33] M. Zeerak Khan, "Automatic-License-Plate-Recognition-using-YOLOv8,” GitHub. [Online]. Available: https://github.com/Muhammad-Zeerak-Khan/Automatic-License-Plate-Recognition-using-YOLOv8. [Accessed: April 29, 2024].